本节内容 最短路径 Floyd算法

Robert W. Floyd



罗伯特·弗洛伊德 (1936-2001) Robert W. Floyd



- Floyd算法 (Floyd-Warshall算法)
- 堆排序算法

最短路径问题



BFS算法(无权图)

单源最短路径

Dijkstra 算法(带权图、无权图)

各顶点间的最短路径

Floyd 算法(带权图、无权图)





Floyd算法:求出每一对顶点之间的最短路径

使用动态规划思想,将问题的求解分为多个阶段

对于n个顶点的图G,求任意一对顶点 Vi —> Vj 之间的最短路径可分为如下几个阶段:

#初始:不允许在其他顶点中转,最短路径是?

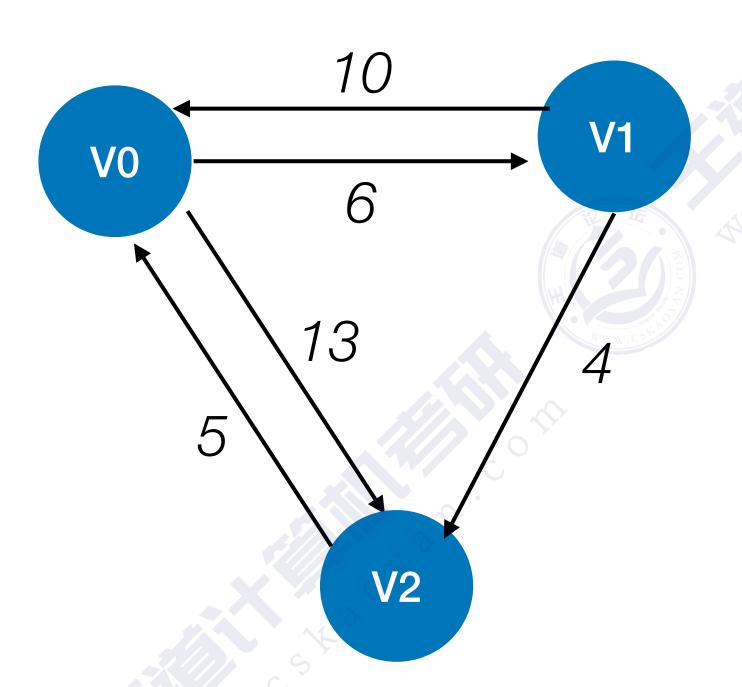
#0: 若允许在 Vo 中转,最短路径是?

#1: 若允许在 Vo、V1 中转, 最短路径是?

#2: 若允许在 Vo、V1、V2 中转,最短路径是?

...

#n-1: 若允许在 Vo、V1、V2..... Vn-1 中转, 最短路径是?



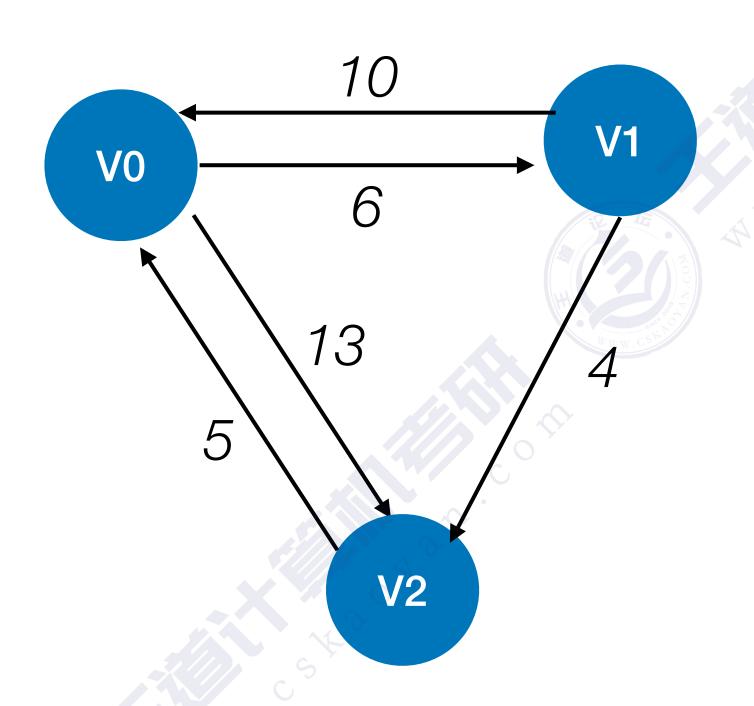


	VO	V1	V2
VO	0	6	13
V1	10	0	4
V2	5	∞	0



	VO	V1	V2
VO	-1	-1.0	-1
V1	-1	-1	-1
V2	1-1	-1	-1

#初始: 不允许在其他顶点中转, 最短路径是?





	VO	V1	V2
VO	0	6	13
V1	10	0	4
V2	5	∞	0

SINCE	两个顶点之 间的中转点	
	path ⁽⁻¹⁾ :	l/so

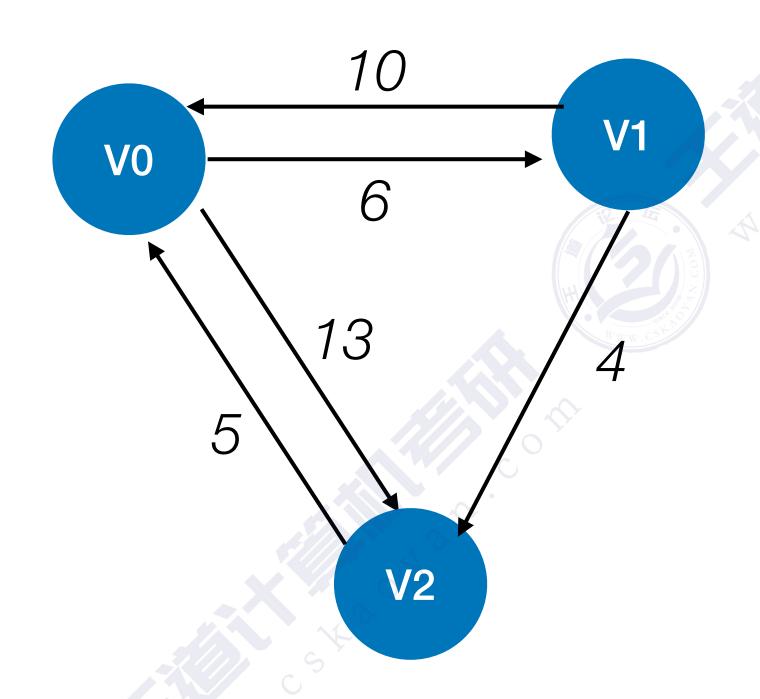
6/1/2				
	VO	V1	V2	
VO	-1	-1.0	-1	
V1	-1	-1	-1	
V2	-1	-1	-1	

#0: 若允许在 Vo 中转,最短路径是? ——求 A⁽⁰⁾和 path⁽⁰⁾

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$ path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$$A^{(-1)}[2][1] > A^{(-1)}[2][0] + A^{(-1)}[0][1] = 11$$

 $A^{(0)}[2][1] = 11$
 $path^{(0)}[2][1] = 0;$



目前来看,各 顶点间的最短 路径长度

 $A^{(-1)} =$

	VO	V1	V2
VO	0	6	13
V1	10	0	4
V2	5	∞	0

两个顶点之间的中转点

path(-1) =

	VO	V1	V2
VO	-1	-1.0	-1
V1	-1	-1	-1
V2	/ -1	-1	-1

#0: 若允许在 Vo 中转,最短路径是? ——求 A⁽⁰⁾和 path⁽⁰⁾

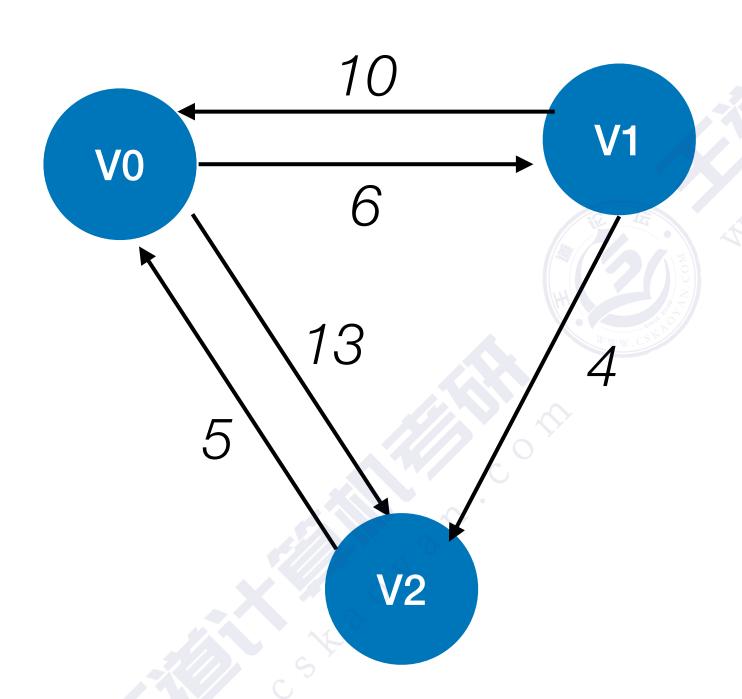
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

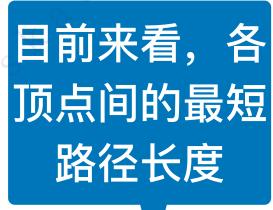
A(0) =

	V0	V1	V2
VO	0	6	13
V1	10	0	4
V2	5	11	0

path⁽⁰⁾ =

	VO	V1	V2
VO	-1	-1	-1
V 1	-1	-1	-1
V2	-1	0	-1





 $A^{(0)} =$

	VO	V1	V2
VO	0	6	13
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

path(0) =

	VO	V1	V2
VO	-1	_1.°	-1
V1	15	-1	-1
V2	-1	0	-1

#1: 若允许在 Vo、 V1中转,最短路径是? ——求 A(1) 和 path(1)

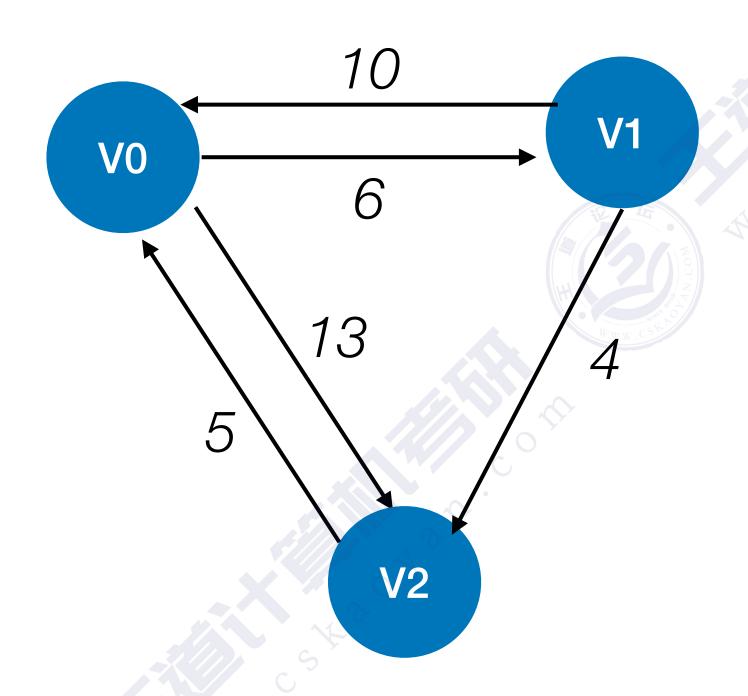
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$ path^(k)[i][j] = k

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

 $A^{(0)}[0][2] > A^{(0)}[0][1] + A^{(0)}[1][2] = 10$

 $A^{(1)}[0][2] = 10$

 $path^{(1)}[0][2] = 1;$



目前来看,各 顶点间的最短 路径长度

 $A^{(0)} =$

	VO	V1	V2
VO	0	6	13
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

path(0) =

	VO	V1	V2
VO	-1	-1 .	-1
V1	-14	-1	-1
V2	/ _1	0	-1

#1: 若允许在 Vo、 V1中转,最短路径是? ——求 A(1) 和 path(1)

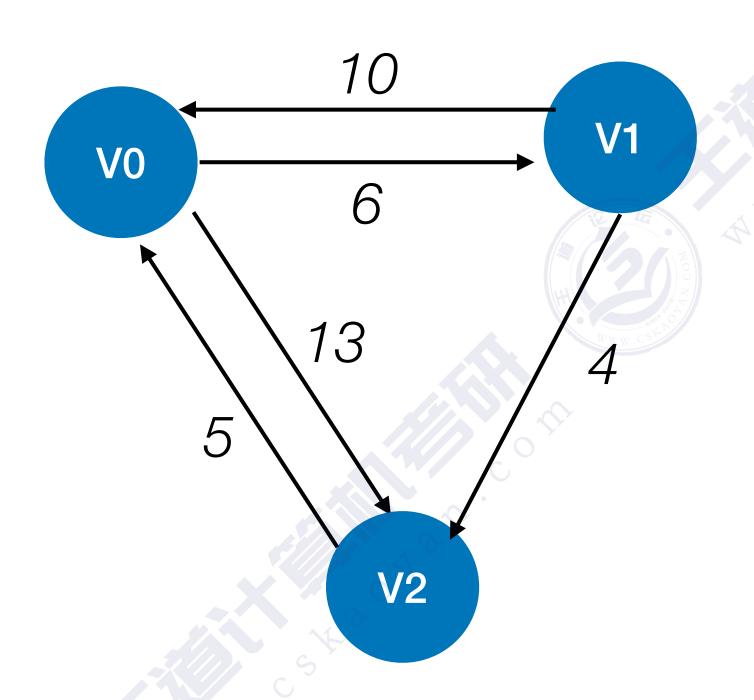
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

A(1) =

		VO	V1	V2
I	VO	0	6	10
	V1	10	0	4
	V2	5	11	0

path(1) =

	VO	V1	V2
VO	-1	-1	1
V1	-1	-1	-1
V2	-1	0	-1





 大度
 V0
 V1
 V2

 $A^{(1)} =$ V0
 0
 6
 10

 V1
 10
 0
 4

 V2
 5
 11
 0

两个顶点之间的中转点 path⁽¹⁾ =

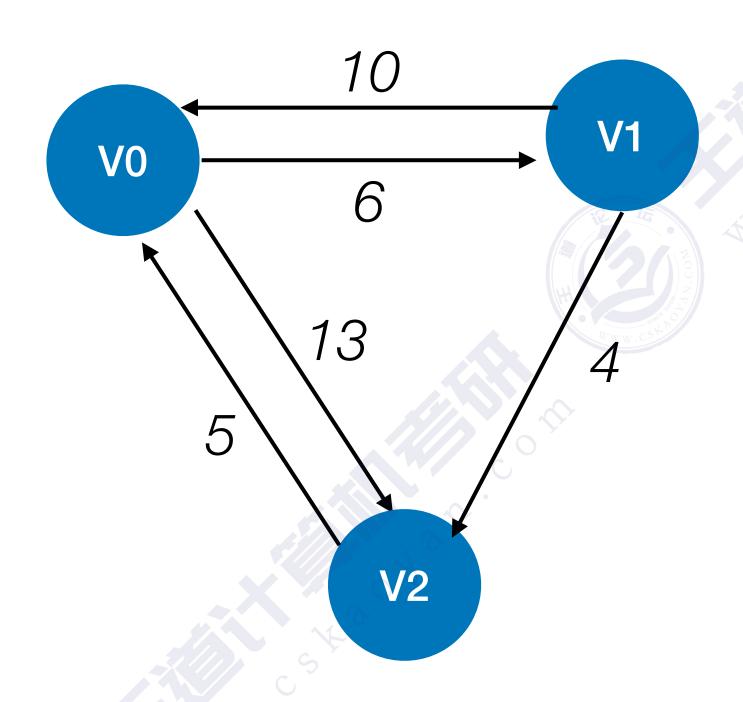
		VO	V1	V2
	VO	-1	-1.°	1
01/	V1	14	-1	-1
A CONTRACTOR OF THE PARTY OF TH	V2	1	0	-1

#2: 若允许在 V₀、V₁、V₂中转,最短路径是? ——求 A⁽²⁾和 path⁽²⁾

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$ $path^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

$$A^{(1)}[1][0] > A^{(1)}[1][2] + A^{(1)}[2][0] = 9$$

 $A^{(2)}[1][0] = 9$
 $path^{(2)}[1][0] = 2;$



目前来看,各 顶点间的最短 路径长度

 $A^{(1)} =$

	VO	V1	V2
VO	0	6	10
V1	10	0	4
V2	5	11	0

两个顶点之间的中转点

path⁽¹⁾ =

	VO	V1	V2
VO	-1	_1.°	1
V1	-1	-1	-1
V2	_1	0	-1

#2: 若允许在 V₀、V₁、V₂中转,最短路径是? ——求 A⁽²⁾和 path⁽²⁾

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path^(k)[i][j] = k 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

A(2) =

		V0	V1	V2
	VO	0	6	10
=	V1	9	0	4
	V2	5	11	0

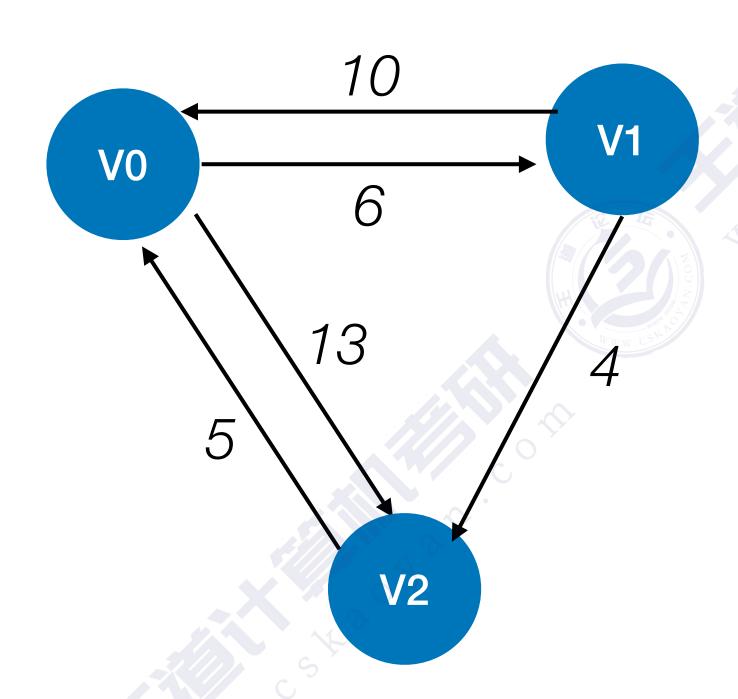
path⁽²⁾ =

 V0
 V1
 V2

 V0
 -1
 -1
 1

 V1
 2
 -1
 -1

 V2
 -1
 0
 -1



目前来看,各 顶点间的最短 路径长度

 $A^{(2)} =$

	VO	V1	V2
VO	0	6	10
V1	9	0	4
V2	5	11	0

两个顶点之间的中转点

path⁽²⁾ =

		VO	V1	V2	
	VO	-1	<u>-</u> 1.°	1	
	V1	2	-1	-1	
	V2	1-1	0	-1	

从A⁽⁻¹⁾和 path⁽⁻¹⁾开始,经过 n 轮递推,得到 A⁽ⁿ⁻¹⁾和 path⁽ⁿ⁻¹⁾

根据 A⁽²⁾ 可知, V1到V2 最短路径长度为 4, 根据 path⁽²⁾ 可知, 完整路径信息为 V1_V2

根据 A⁽²⁾ 可知, VO到V2 最短路径长度为 10, 根据 path⁽²⁾ 可知, 完整路径信息为 VO_V1_V2

根据 A⁽²⁾ 可知, V1到V0 最短路径长度为 9, 根据 path⁽²⁾ 可知, 完整路径信息为 V1_V2_V0

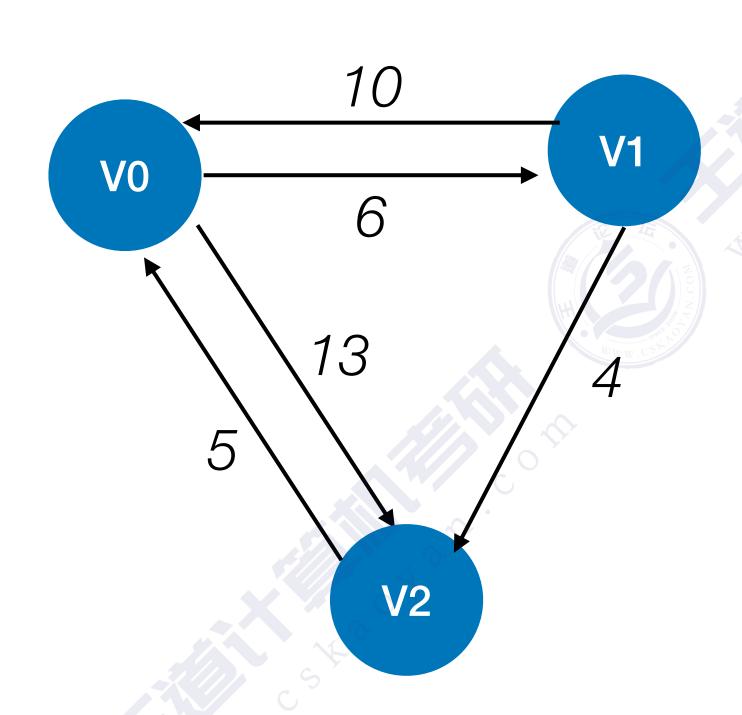
若则

 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$

 $path^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

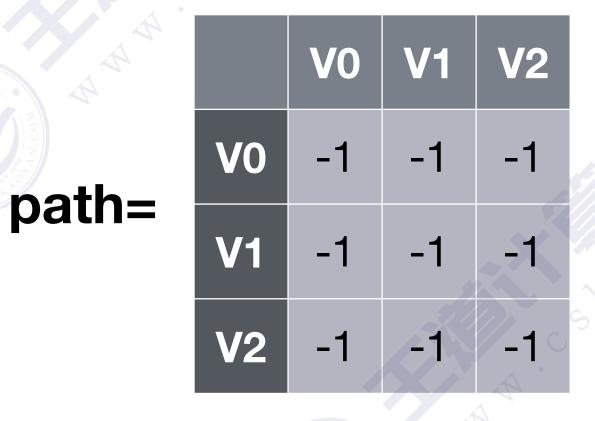
王道考研/CSKAOYAN.COM

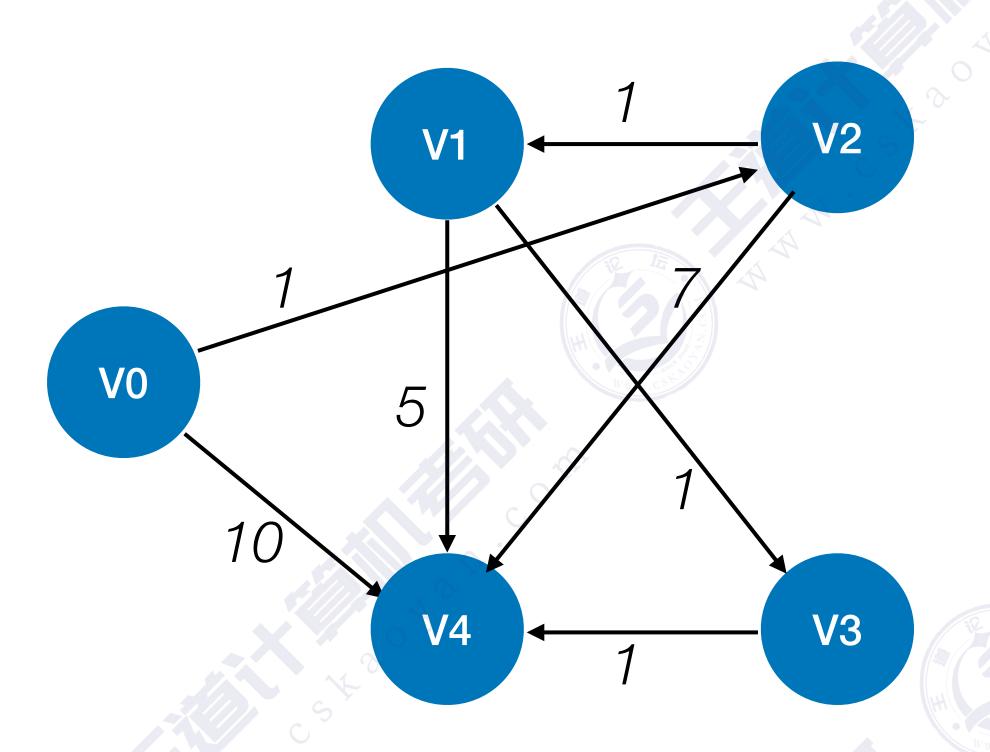
Floyd算法核心代码

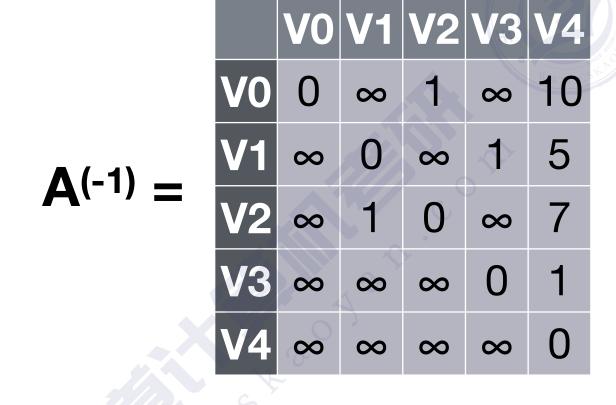


若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

		VO	V1	V2
A —	VO	0	6	13
A =	V1	10	0	4
	V2	5	∞	0

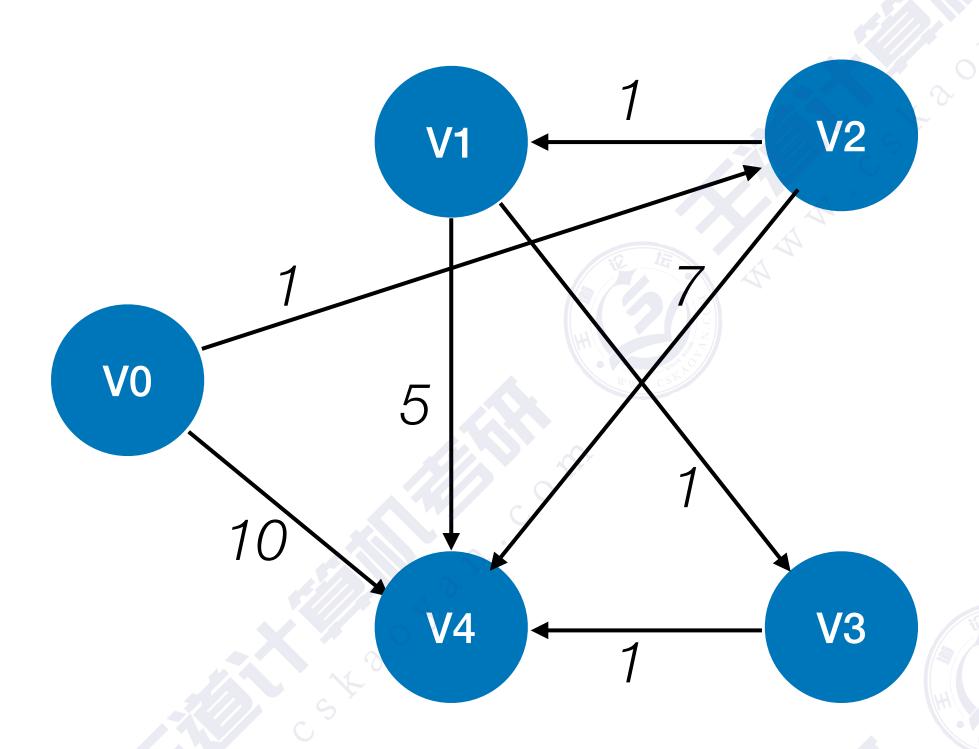




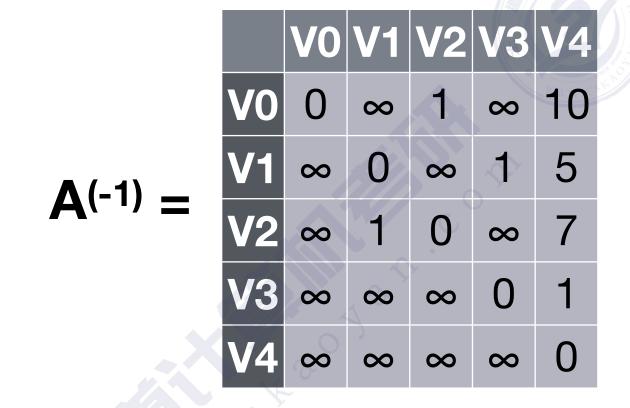


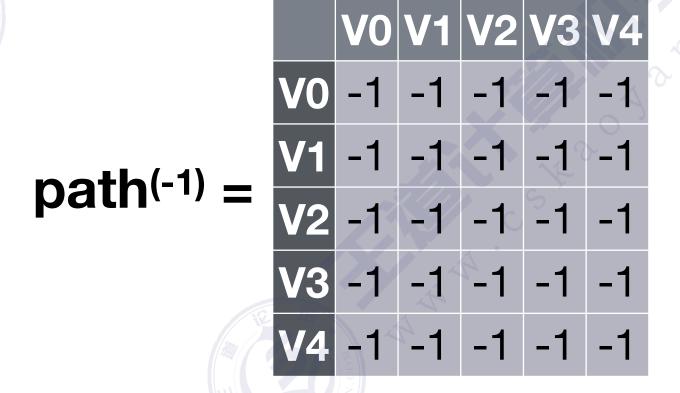
		VO	V 1	V2	V3	V4
	VO	-1	-1	-1	-1	-1
noth(-1) _	V1	-1	-1	-1	12	-1
path ⁽⁻¹⁾ =	V2	-1	-1	-10	-1	-1
	V 3	-1	-1	-1	-1	-1
	V 4	1	-1	-1	-1	-1

#初始: 不允许在其他顶点中转, 最短路径是?

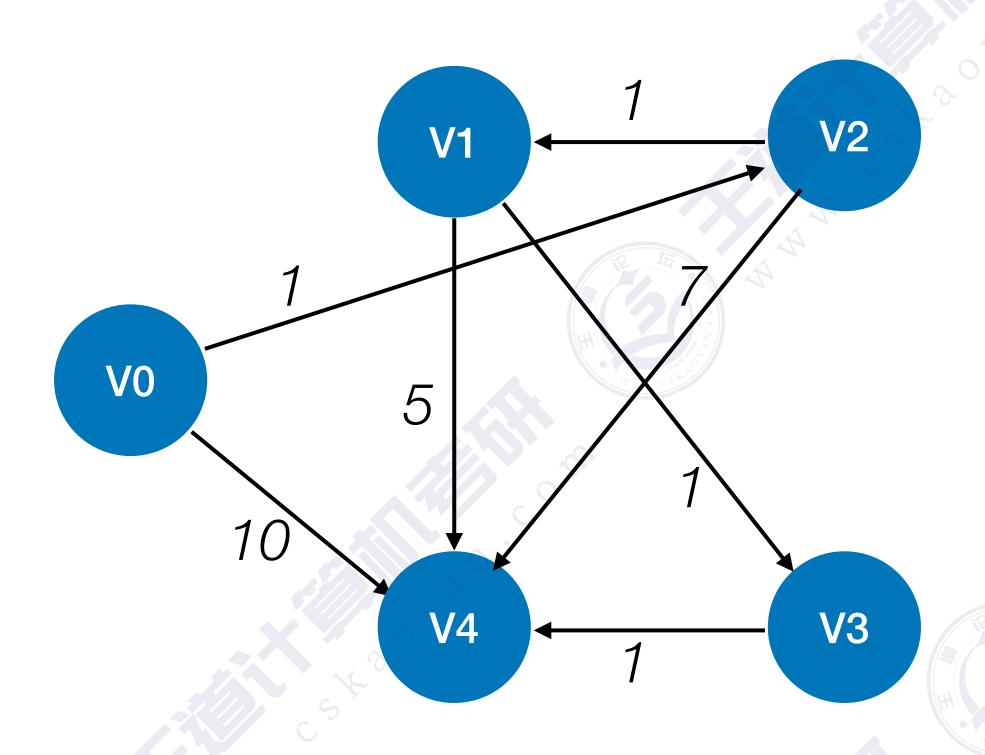


若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值





#0: 若允许在 Vo 中转,最短路径是? ——求 A⁽⁰⁾和 path⁽⁰⁾

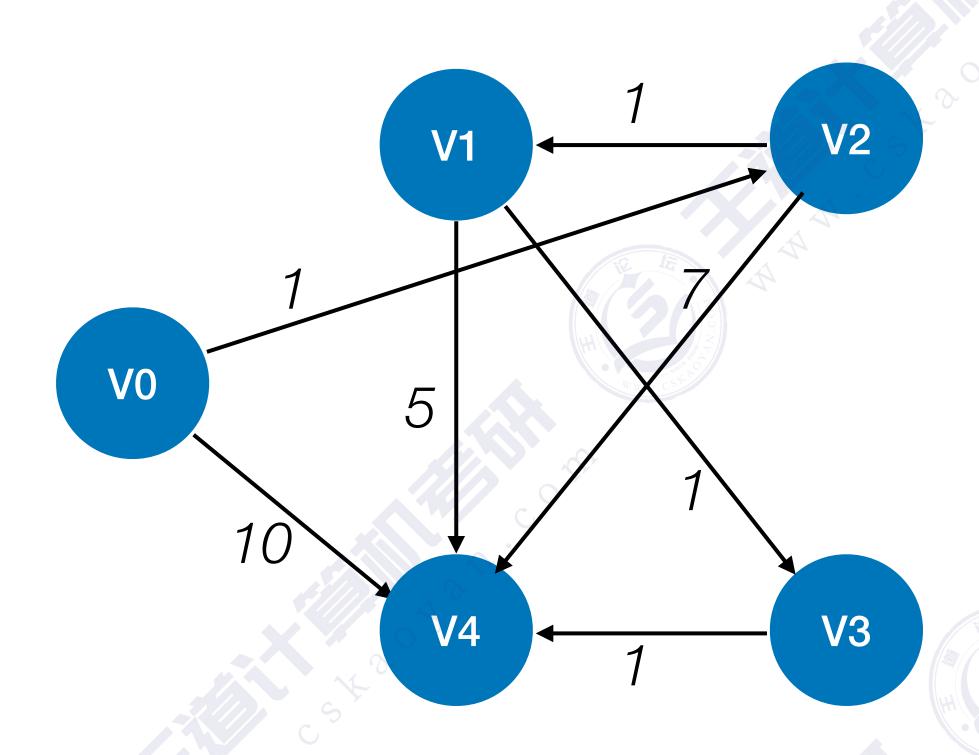


		VO	V 1	V2	V 3	V 4
	VO	0	∞	1	∞	10
$A^{(-1)} =$	V1	∞	0	∞		5
A(') =	V2	∞	1	0	∞	7
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

#0: 若允许在 Vo 中转,最短路径是? ——求 A⁽⁰⁾和 path⁽⁰⁾

若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

	VO	-1	-1	-1	-1	-1	
path ⁽⁰⁾ =	V1	-1	-1	-1	-1	-1	
paun ^w =	V2	-1	-1	-1	-1	-1	
	V 3	-1	-1	-1	-1	-1	
	V 4	-1	-1	-1	-1	-1	

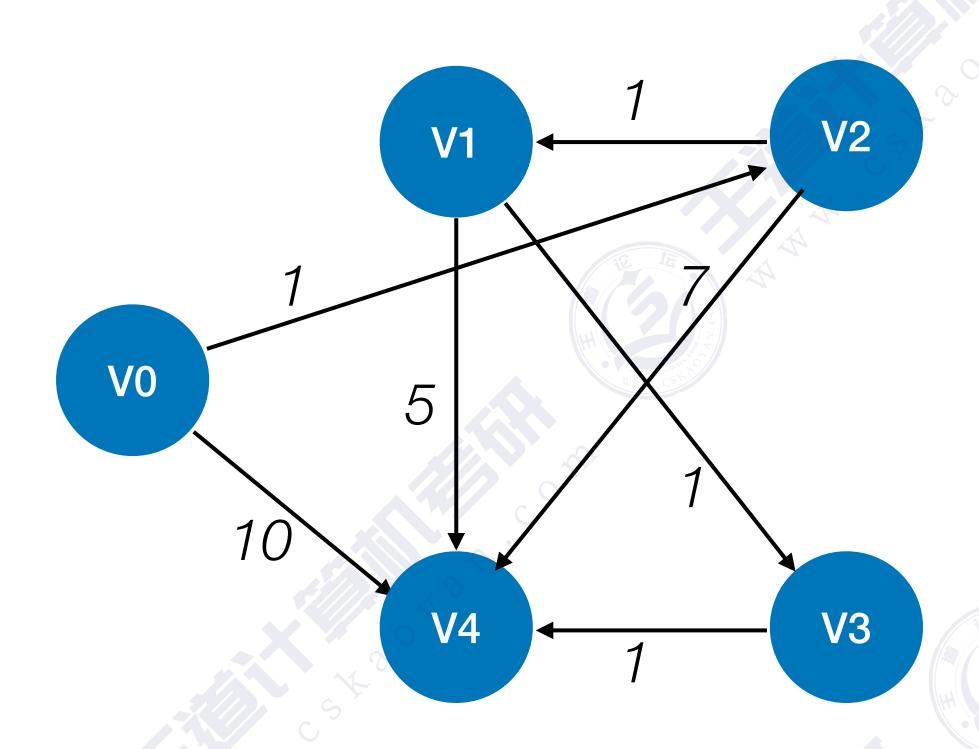


若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

		VO	V1	V2	V 3	V4
	VO	0	∞	1	∞	10
A (0) =	V1	∞	0	∞		5
	V2	∞	1	0	∞	7
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

#1: 若允许在 Vo、 V1中转,最短路径是? ——求 A(1) 和 path(1)

```
for(int i=0; i<n; i++) { //遍历整个矩阵, i为行号, j为列号
for (int j=0; j<n; j++) {
    if (A[i][j]>A[i][k]+A[k][j]) { //以 Vk 为中转点的路径更短
        A[i][j]=A[i][k]+A[k][j]; //更新最短路径长度
        path[i][j]=k; //中转点
    }
    }
}
```



若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$ path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

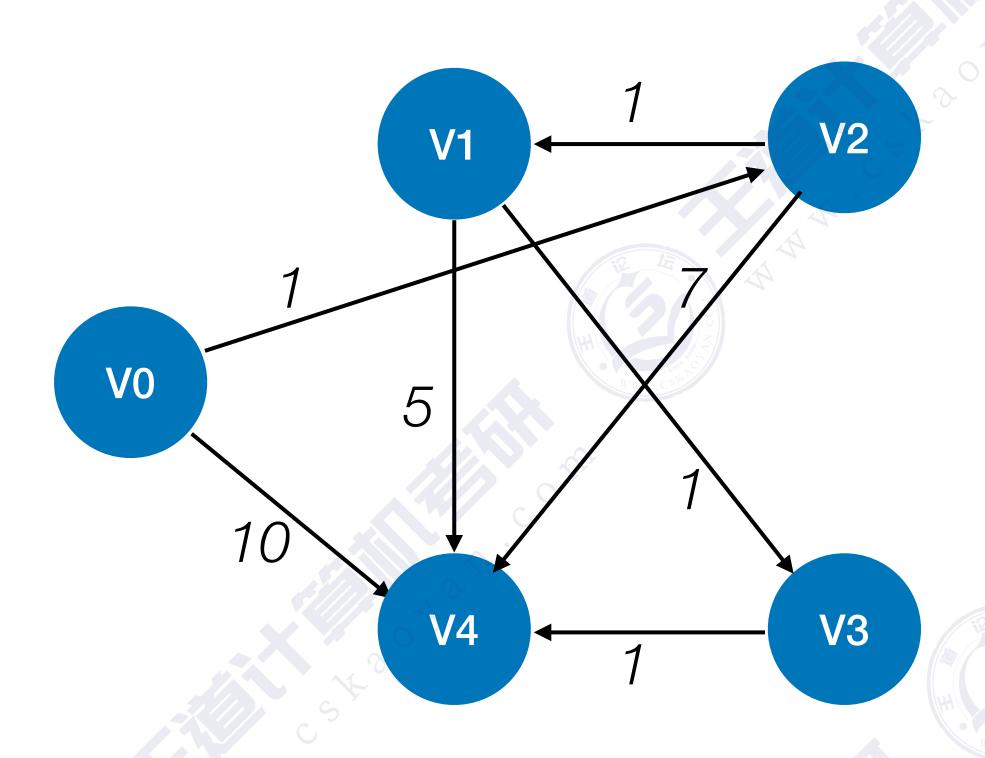
#1: 若允许在 Vo、 V1中转,最短路径是? ——求 A(1)和 path(1)

$$A^{(0)}[2][3] > A^{(0)}[2][1] + A^{(0)}[1][3] = 2$$

 $A^{(1)}[2][3] = 2$
 $path^{(1)}[2][3] = 1;$

$$A^{(0)}[2][4] > A^{(0)}[2][1] + A^{(0)}[1][4] = 6$$

 $A^{(1)}[2][4] = 6$
 $path^{(1)}[2][4] = 1;$

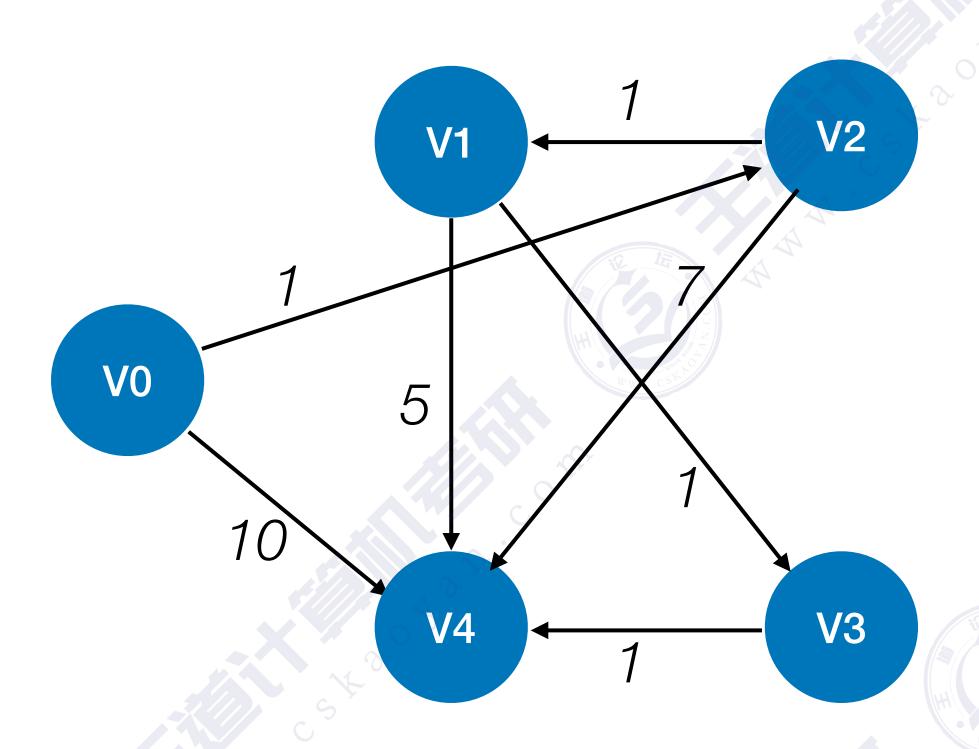


		V0	V 1	V2	V 3	V4
	VO	0	∞	1	∞	10
A (0) —	V1	∞	0	∞	4	5
$A^{(0)} =$	V2	∞	1	0	∞	7
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

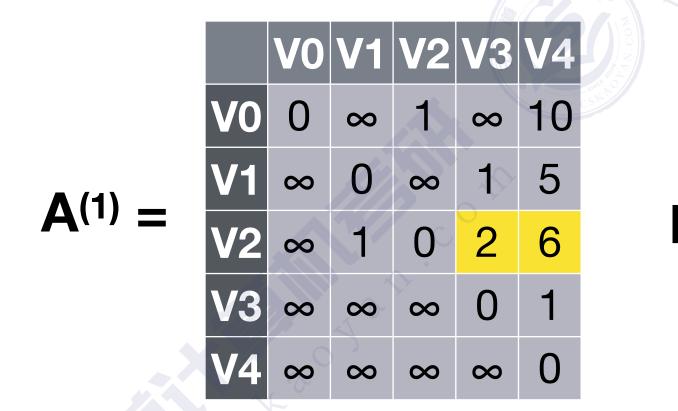
#1: 若允许在 Vo、 V1中转,最短路径是? ——求 A(1) 和 path(1)

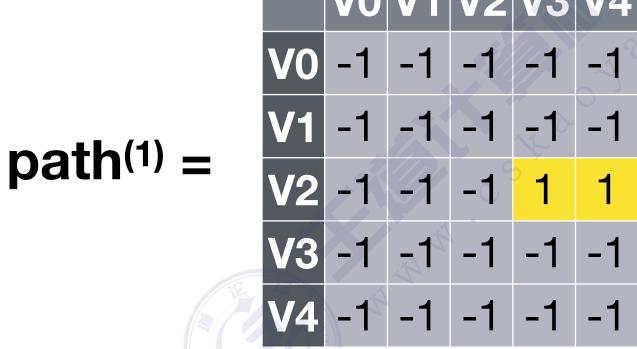
若	$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$
则	$A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$
	$path^{(k)}[i][j] = k$
否则	A(k) 和 path(k) 保持原值

		VU	VI	VZ	VO	V	
	VO	-1	-1	-1	-1	-1	
	V1	-1	-1	-1	-1	-1	
path ⁽¹⁾ =	V2	-1	-1	-1	1	1	
	V 3	-1	-1	-1	-1	-1	
	V 4	-1	-1	-1	-1	-1	

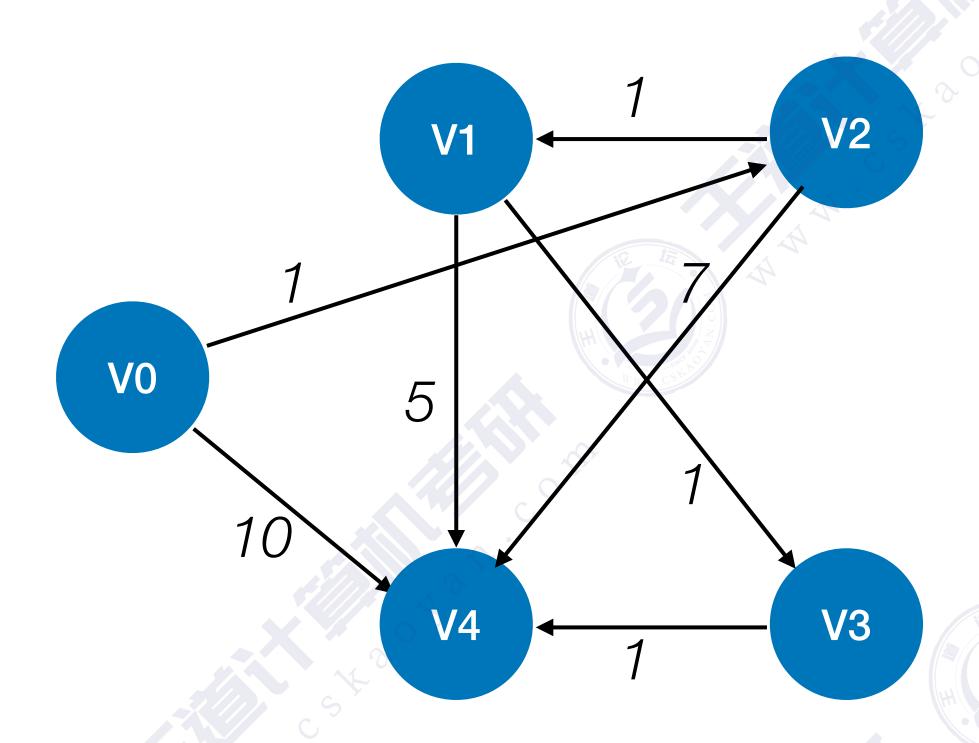


若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值



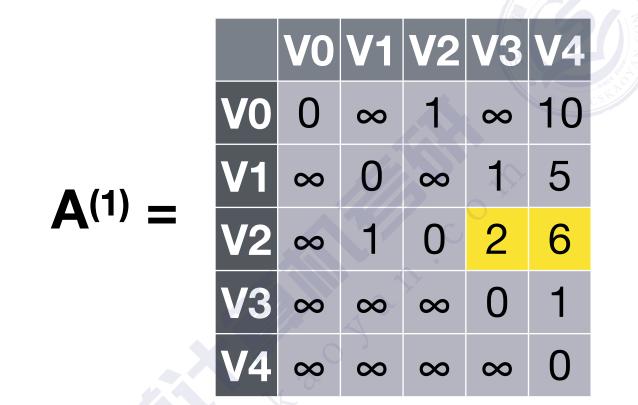


#2: 若允许在 V₀、V₁、V₂中转,最短路径是? ——求 A⁽²⁾和 path⁽²⁾



若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$ path(k)[i][j] = k不可 A(k) 和 path(k) 包共百角

否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值



#2: 若允许在 V₀、V₁、V₂中转,最短路径是? ——求 A⁽²⁾和 path⁽²⁾

$$A^{(1)}[0][1] > A^{(1)}[0][2] + A^{(1)}[2][1] = 2$$

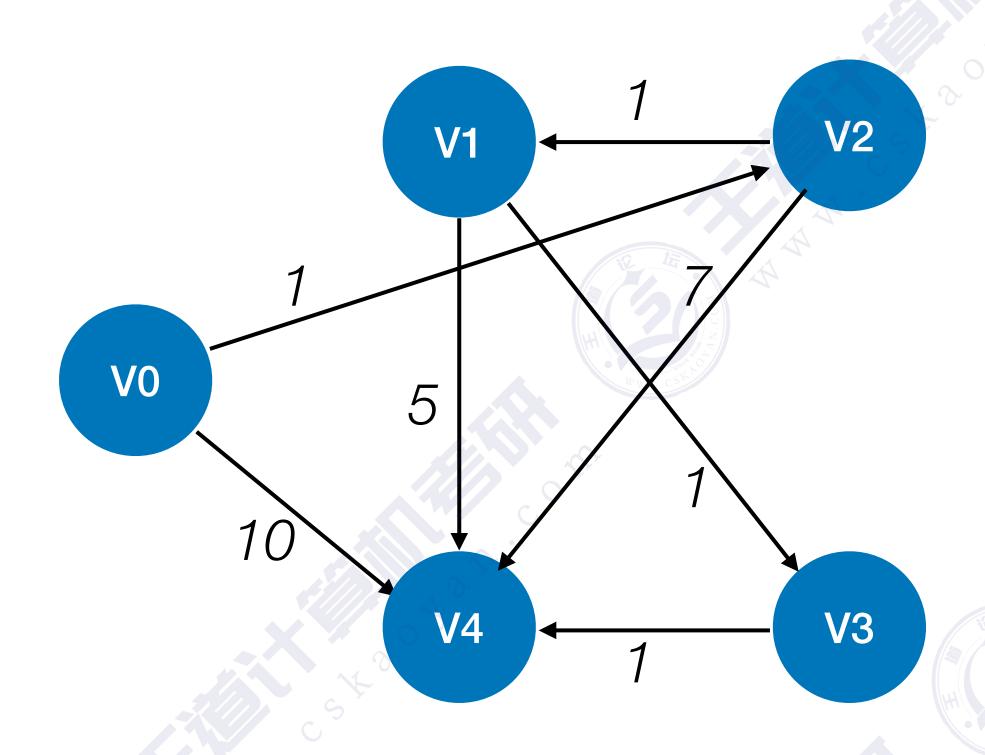
 $A^{(2)}[0][1] = 2$; path⁽²⁾[0][1] = 2;

$$A^{(1)}[0][3] > A^{(1)}[0][2] + A^{(1)}[2][3] = 3$$

$$A^{(2)}[0][3] = 3$$
; path $^{(2)}[0][3] = 2$;

$$A^{(1)}[0][4] > A^{(1)}[0][2] + A^{(1)}[2][4] = 7$$

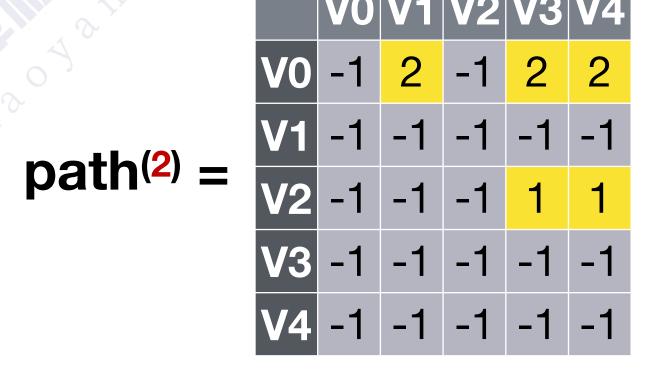
 $A^{(2)}[0][4] = 7$; path⁽²⁾[0][4] = 2;

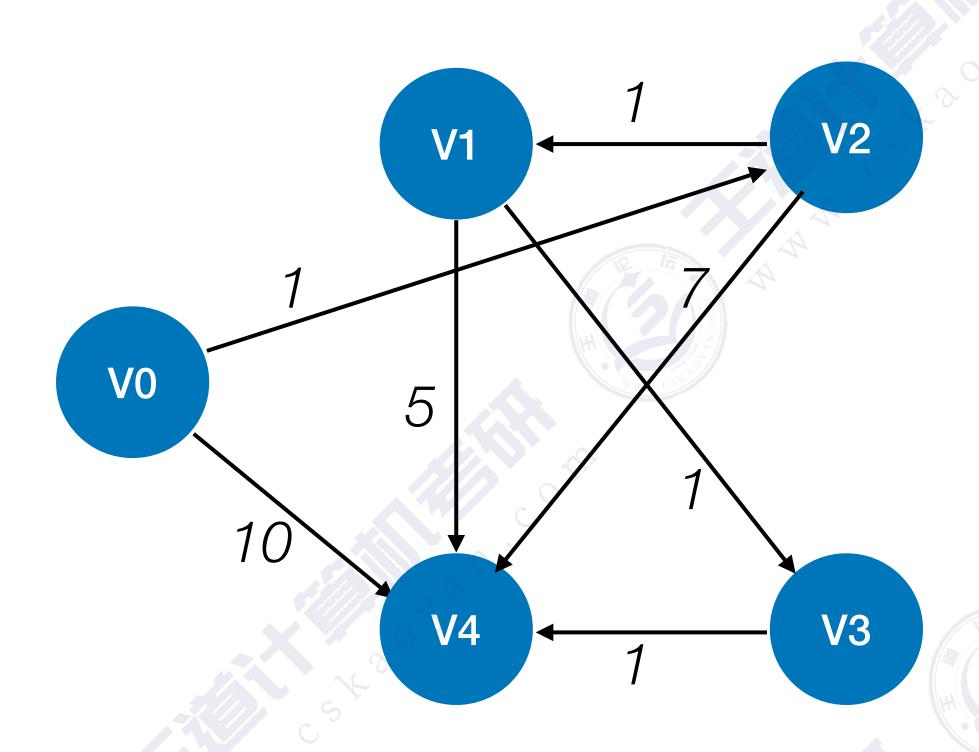


		VO	V1	V2	V 3	V4
A (1) =	VO	0	∞	1	∞	10
	V1	∞	0	∞	15	5
	V2	∞	1	0	2	6
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

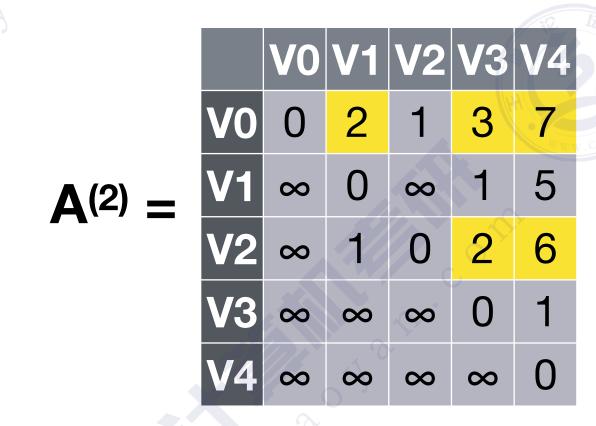
#2: 若允许在 V₀、V₁、V₂中转,最短路径是? ——求 A⁽²⁾和 path⁽²⁾

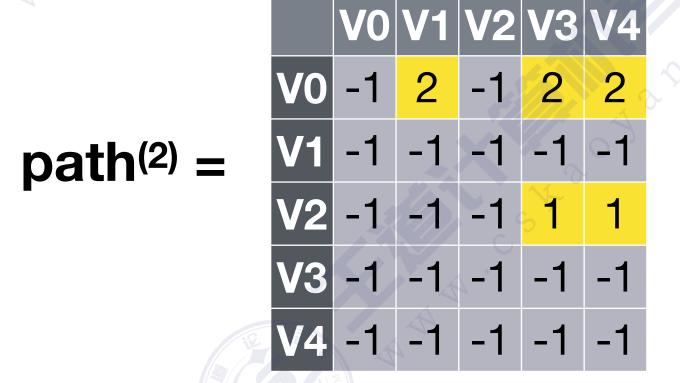
若
$$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$$
 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值





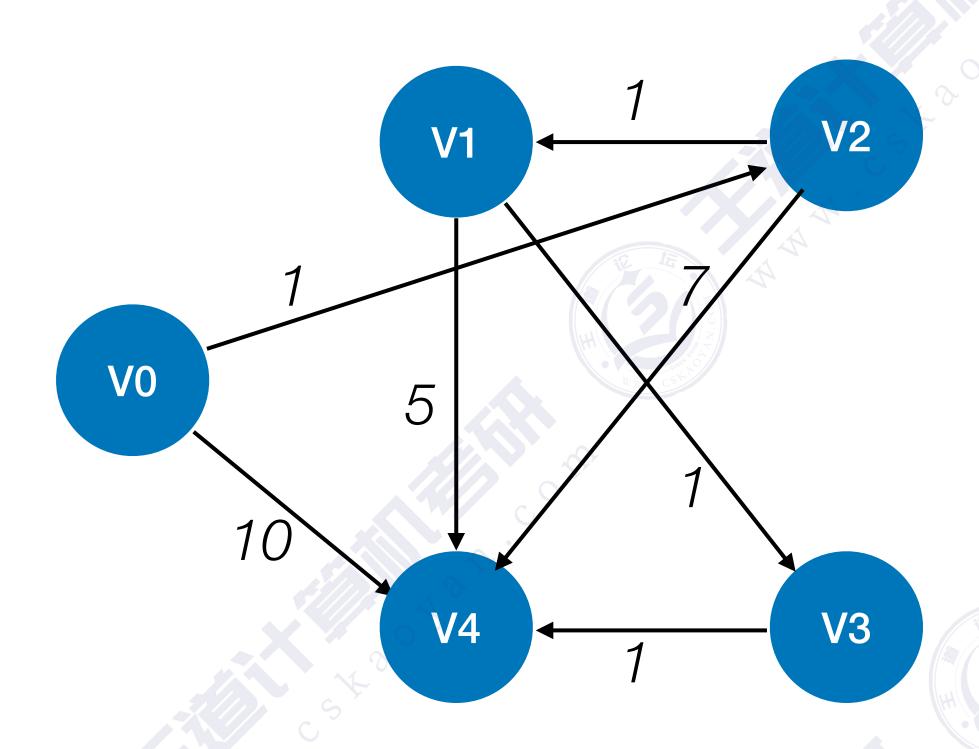
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值





#3: 若允许在 V₀、V₁、V₂、V₃中转,最短路径是? ——求 A⁽³⁾和 path⁽³⁾

```
for(int i=0; i<n; i++) { //遍历整个矩阵, i为行号, j为列号
for (int j=0; j<n; j++) {
    if (A[i][j]>A[i][k]+A[k][j]) { //以 Vk 为中转点的路径更短
        A[i][j]=A[i][k]+A[k][j]; //更新最短路径长度
        path[i][j]=k; //中转点
    }
}
```



若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j];$ path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

#3: 若允许在 V₀、V₁、V₂、V₃中转,最短路径是? ——求 A⁽³⁾ 和 path⁽³⁾

$$A^{(2)}[0][4] > A^{(2)}[0][3] + A^{(2)}[3][4] = 4$$

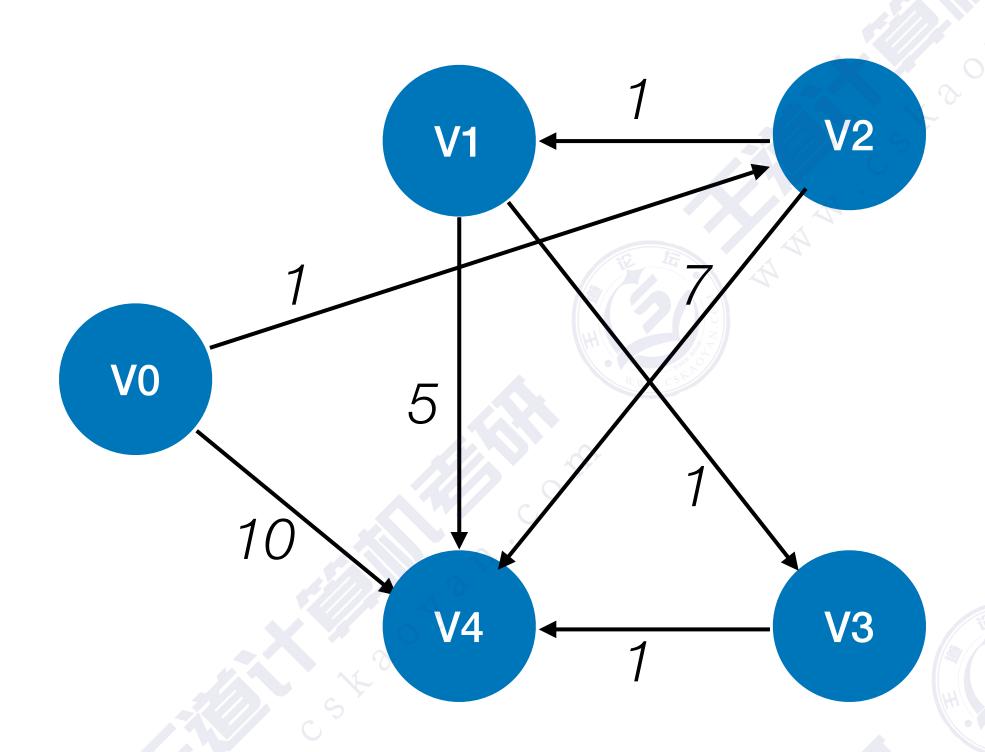
 $A^{(3)}[0][4] = 4$; path⁽³⁾[0][4] = 3;

$$A^{(2)}[1][4] > A^{(2)}[1][3] + A^{(2)}[3][4] = 2$$

$$A^{(3)}[1][4] = 2$$
; path $^{(3)}[1][4] = 3$;

$$A^{(2)}[2][4] > A^{(2)}[2][3] + A^{(2)}[3][4] = 3$$

$$A^{(3)}[2][4] = 3$$
; path $^{(3)}[2][4] = 3$;

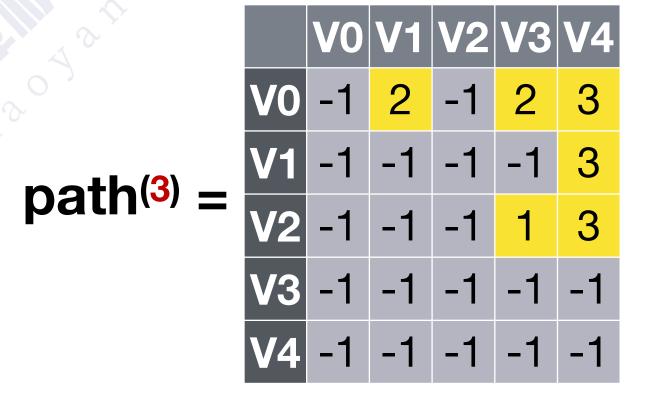


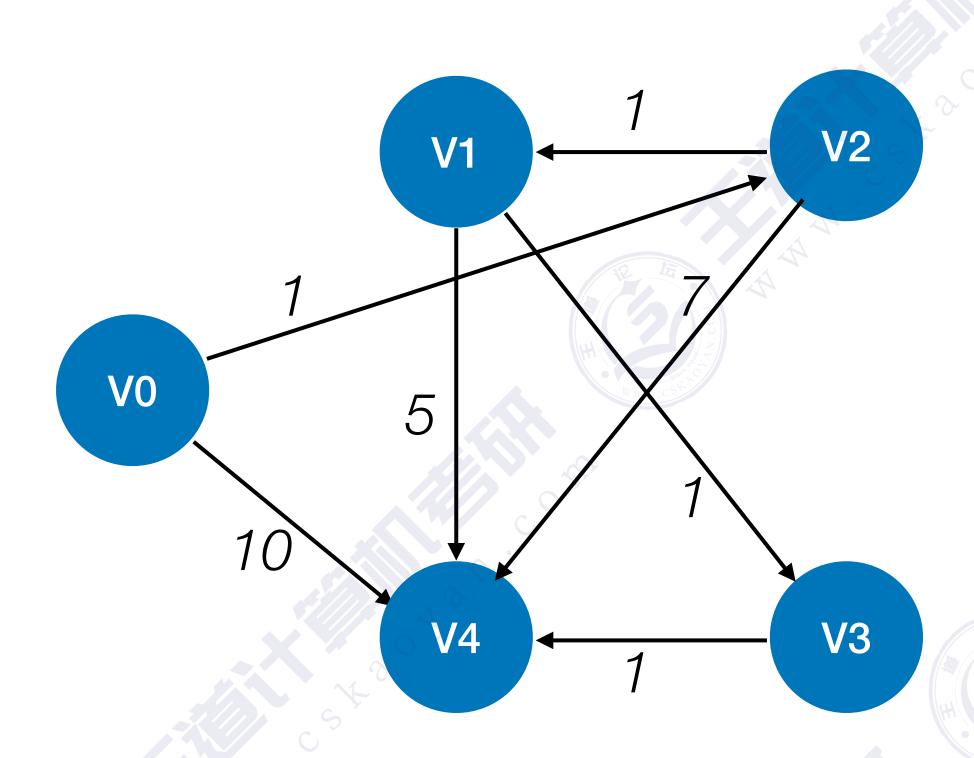
		VO	V1	V2	V3	V 4
	VO	0	2	1	3	7
$\mathbf{A}^{(2)} =$	V1	∞	0	∞	1	5
	V2	∞	1	0	2	6
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

#3: 若允许在 V₀、V₁、V₂、V₃中转,最短路径是? ——求 A⁽³⁾ 和 path⁽³⁾

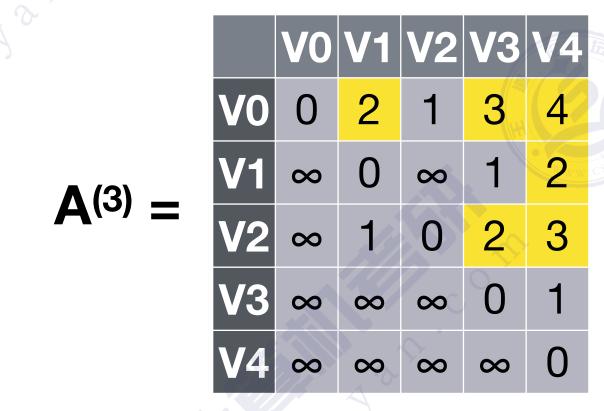
若
$$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$$

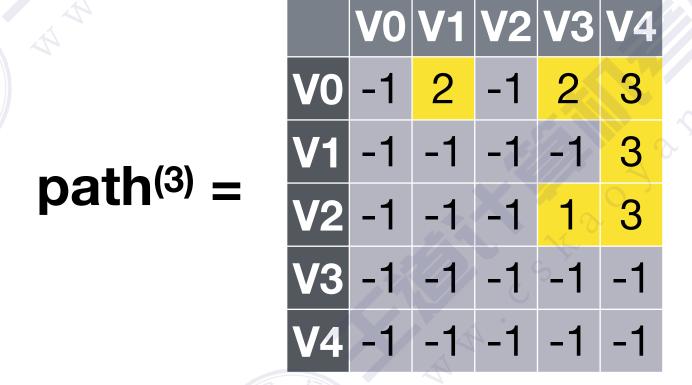
则 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$;
path $^{(k)}[i][j] = k$
否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值





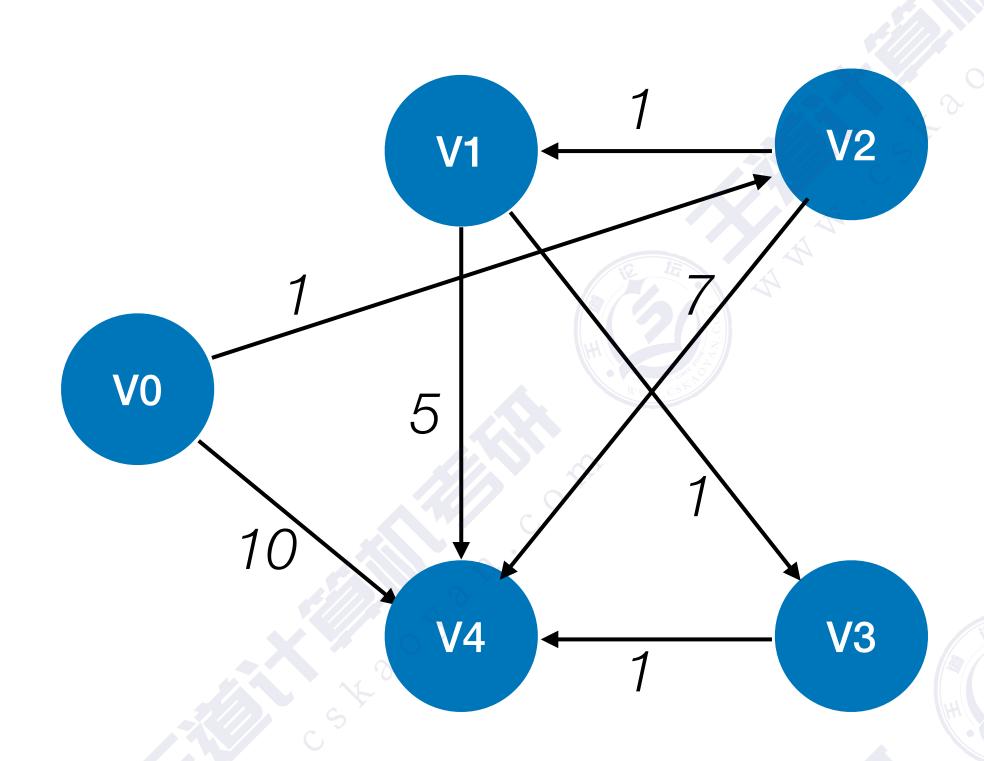
若 $A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$ 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值





#4: 若允许在 V₀、V₁、V₂、V₃、V₄中转,最短路径是? ——求 A⁽⁴⁾和 path⁽⁴⁾

```
for(int i=0; i<n; i++) { //遍历整个矩阵, i为行号, j为列号
for (int j=0; j<n; j++) {
    if (A[i][j]>A[i][k]+A[k][j]) { //以 Vk 为中转点的路径更短
        A[i][j]=A[i][k]+A[k][j]; //更新最短路径长度
        path[i][j]=k; //中转点
    }
    }
}
```



		VO	V1	V2	V 3	V 4
	VO	0	2	1	3	4
$A^{(3)} =$	V1	∞	0	∞	1	2
	V2	∞	1	0	2	3
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

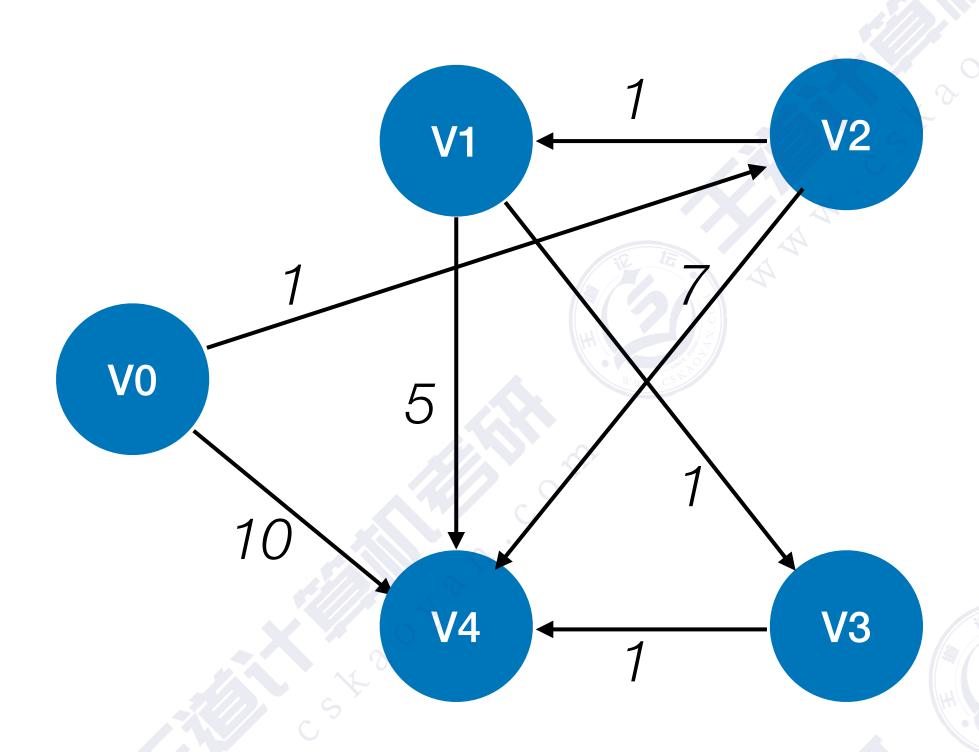
		VU	VI	VZ	VO	V4
	VO	-1	2	-1	2	3
Ll ₂ (3)	V1	-1	-1	-1	-1	3
path ⁽³⁾ =	V2	-1	-1	-1	10	3
	V 3	-1	-1	-1	-1	-1
	V4	-1	-14	-1	-1	-1

#4: 若允许在 V₀、V₁、V₂、V₃、V₄中转,最短路径是? ——求 A⁽⁴⁾和 path⁽⁴⁾

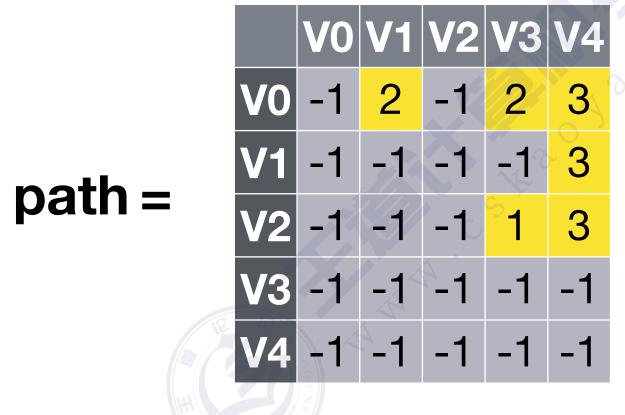
若
$$A^{(k-1)}[i][j] > A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$$
 列 $A^{(k)}[i][j] = A^{(k-1)}[i][k] + A^{(k-1)}[k][j]$; path $^{(k)}[i][j] = k$ 否则 $A^{(k)}$ 和 $path^{(k)}$ 保持原值

		VO	V1	V2	V 3	V4
	VO	0	2	1	3	4
A (4)	V1	∞	0	∞	1	2
$A^{(4)} =$	V2	∞	1	0	2	3
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

		VU	VI	VZ	VJ	V4
	VO	-1	2	-1	2	3
o o t b (4)	V1	-1	-1	-1	-1	3
path ⁽⁴⁾ =	V2	-1	-1	-1	1	3
	V 3	-1	-1	-1	-1	-1
	V 4	-1	-1	-1	-1	-1



		VO	V1	V2	V 3	V 4
	VO	0	2	1	3	4
A —	V1	∞	0	∞	10	2
	V2	∞	1	0	2	3
	V 3	∞	∞	∞	0	1
	V 4	∞	∞	∞	∞	0

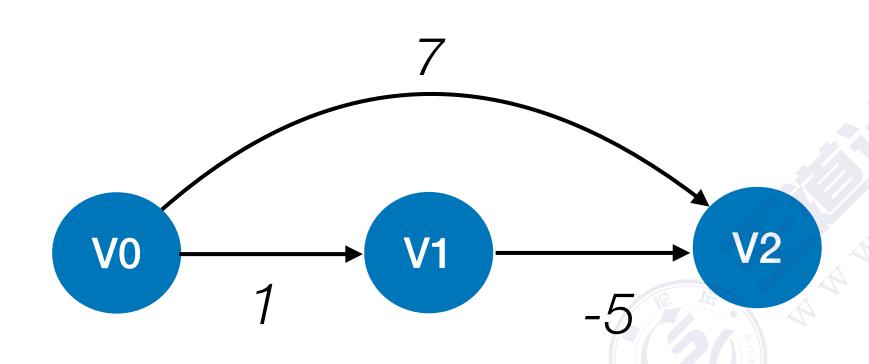


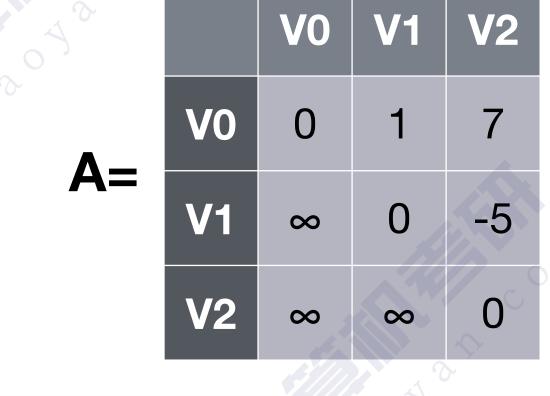
VO到V4 最短路径长度为 A[0][4]=4

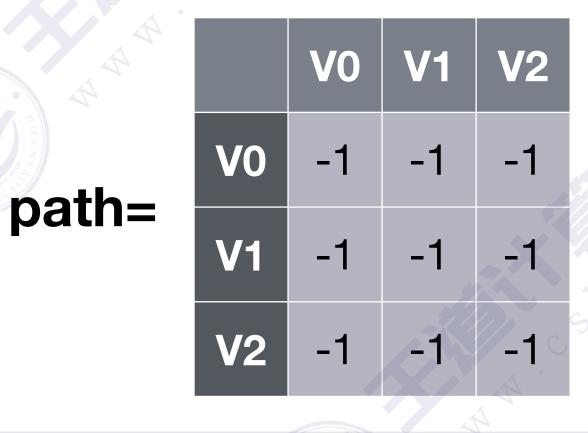
通过path矩阵递归地找到完整路径:

V0		V3	V4
V 0	V2	V3	V 4
V0	V2	V1 V3	V 4

练习: Floyd算法用于负权图









Floyd算法可以用于 负权值带权图

不能解决的问题 -9 **V2**



知识点回顾与重要考点

		BFS 算法	Dijkstra 算法	Floyd 算法
	无权图		Www.cs*	
	带权图	X		√
	带负权值的图	X	X	
	带负权回路的图	X		Www.cst.
	时间复杂度	O(V ²)或O(V + E)	O(V ²)	O(V ³)
	通常用于	求无权图的单源最 短路径	求带权图的单源最 短路径	求带权图中各顶点 间的最短路径

注: 也可用 Dijkstra 算法求所有顶点间的最短路径,重复 |V| 次即可,总的时间复杂度也是O(|V|³)